In the United States Patent and Trademark Office

In re the Application of:

Jason M. Bell	)	
Serial Number: 10/809,583	)	Group: 2167
Docket Number: AUS920040052US1	)	Examiner: Kimberly M. Lovel
Filed on: 03/25/2004	)	
For: "Real-time Attribute Processor and Syntax Schema for Directory Access Protocol Services"	)	

APPEAL BRIEF

Real Party in Interest per 37 CFR §41.37(c)(1)(i)

The subject patent application is owned by International Business Machines Corporation of Armonk, NY.

Related Appeals and Interferences per 37 CFR §41.37(c)(1)(ii)

None.

Status of Claims per 37 CFR §41.37(c)(1)(iii)

Claims 8, 12, 13, and 20 - 34 are finally rejected, from which the present Appeal arises. Claims 1 - 7, 9 - 11, and 14 - 19 were previously cancelled in this present patent application.

Status of Amendments after Final Rejections per 37 CFR §41.37(c)(1)(iv)

No amendments to the claims have been made after the final rejections.

Summary of the Claimed Subject Matter per 37 CFR §41.37(c)(1)(v)

Lightweight Directory Access Protocol (LDAP) is a very popular method of providing a single interface to multiple databases, where the databases often are in different formats and are even on different networked computers. Through use of LDAP, data is accessed the same way regardless of what type of database stores it or where it is stored. LDAP also virtually integrates these multiple database into what appears to be a single database, so that entities reading or storing data into the LDAP structure need not know the location of the actual, real database.

However, LDAP is also known to require considerable processing time to update stored values (e.g. write new values where old ones are already stored). Thus, storing data which is prone to frequent changes in an LDAP database can cause considerable loading on an LDAP server, and often results in slowed access to the databases.

Normally, according to the prior art, data values which are dynamic in nature, such as stock prices, current weather conditions, etc., are periodically updated in the LDAP, which requires the dynamic value to be converted to a static value and then the LDAP "update" command to be processed. This traditional operation can consume considerable server time if there are a lot of dynamic values mapped to the LDAP directory, and/or if there are a lot of read requests for those same values. Another problem is that a client reading one of these values gets a stale or old value (old as the last update cycle).

Appellant's invention reduces the processing load on an LDAP server for requests for values which are dynamic in nature, such as stock prices, news headlines, current weather conditions, etc. Instead of following the traditional process to convert the dynamic value to a static value, then to store it in the LDAP, and then to read it out again upon later request for it, Appellan'ts invention rather provides a special identifier for so-called "real-time" values *associated* with the LDAP, but not stored in the LDAP itself. When a read request is received for one of the specially-identified real-time values as if the requester expects the value to be stored in the LDAP, a special handler detects the special identifier and fetches the real-time value *directly from the source* (e.g. weather.com, stockprices.com, etc.), without reading the value from the LDAP. A first advantage is that this value with be current and not stale, unlike the prior art process which may retrieve a stale or old value.

The special handler then converts the value to an LDAP-message compatible value (e.g. a

static value), and places it in the return message to the requesting client (not in the LDAP directly as the prior art does), thereby shorting and avoiding the steps of storing, updating, and reading the actual LDAP for that particular parameter, and avoiding the inherent server loading and response delays.

Claim 8 is an independent claim directed towards automated methods according to Appellant's invention, the support for which is summarized in the following table.

TABLE - Claim 8 under Appeal

Claim Step	Support in Our Specification and Drawings (Paragraph Numbers as Appearing in Pre-Grant Publication, Appellant's Emphasis added)
8. A method comprising: providing at least one declaration for an attribute to be handled as a real-time attribute associated with but external to a directory structure;	[0076] This example **declaration** for a dynamic attribute . . . declares a unique and previously-registered OID for the example attribute "currentTemp" to be a value which is **not statically stored in the directory**, but which is **to be resolved** in real-time by a user-supplied process, module, method, or function.
receiving a directory access protocol request for access to one or more attribute values from said associated directory structure;	[0090] Turning to FIG. 7, a logic diagram for our enhanced LDAP Server (24') is shown. Two queues (71) are provided to **receive requests** and respond to requests, respectively, for remote users and client applications via a communications network. An LDAP protocol stack and request handler (72) interfaces to the queues (71), and to an LDAP attribute processor. The LDAP attribute processor (73) **examines the requests and the declarations for the attributes** requested, and for the requested attributes which are normal, static LDAP attributes, retrieves or modifies the attribute(s) in the LDAP directory via a bi-directional file system interface (76).

detecting in said received request a request to access an attribute declared as a real-time external attribute;	[0091] However, a new signal, Real-time Attribute Processor ("RTAP) enable (75), is controlled by the LDAP attribute processor (73) to allow for **selective retrieval of values of attributes which are defined by our new real-time data schema**, as previously discussed. When the LDAP attribute processor (73) **receives a request for such a dynamic attribute**, the RTAP enable (75) signal is controlled such that Switch 1 (74) **redirects the data flow from the file system interface (76) to an RTAP Module Selector (78), which then determines which RTAP module (51) to invoke.**
responsive to said detecting of a request for a real-time attribute, resolving a real-time value by obtaining an attribute value from a real-time source external to said directory structure;	[0092] The selected RTAP module (51) is then invoked or addressed, which **accesses one or more real-time data sources** (54) such as web sites, data interfaces, etc., **performs the necessary data manipulations (e.g. calculations, image processing, etc.)**, and returns the resulting data for the real-time value(s) of the attributes to the LDAP attribute process (73) via Switch 1 (74).
responsive to said resolving, converting said obtained attribute value from a real-time attribute to a static attribute, wherein said real-time attribute is incompatible with said directory access protocol, and wherein said static attribute is compatible with said directory access protocol; and	[0087] It is important to note that the v**alue "82 F" was never actually stored in or retrieved from the LDAP directory**. In this manner, the **handling of dynamic data is transparent to any requesting client**, thereby providing a powerful extension to the directory server protocol to allow reading of dynamic data without changing or extending the protocol itself (e.g. client applications are **backwards compatible** to our improved directory server). . . . [0093] In this way, each request is handled by parsing the actions into actions for static attributes, which are handled in the normal LDAP manner, and actions for **dynamic attributes, which are resolved by RTAP modules selected from a group of RTAP modules accordingly**. Dynamic data values, however, are not stored in the LDAP directory or file system, thereby minimizing the overhead impact of adding dynamic data to a directory which is optimized for static data storage. . . . [0137] (e) the interface between the requesting application client and the directory server is unchanged, allowing **backwards compatibility** with legacy applications and protocol stacks; and . . .

returning to a requester said converted real-time attribute directly in said directory access protocol, wherein storing and updating of said converted real-time attribute value in said directory structure is eliminated or avoided.	[0076] . . .which is **not statically stored in the directory**, but which is to be resolved in real-time by a user-supplied process, module, method, or function. . . . [0087] . . . **never actually stored** in or retrieved from the LDAP directory. . . . . . . [0101] (e) **returning (85) the resolved value** to the requesting client **while suppressing or avoiding the storing of the actual value in the LDAP directory;**

Claim 20 is directed towards analogous elements and limitations of embodiments in computer readable memory:

20. A computer readable memory comprising:

a computer readable memory suitable for encoding computer programs; and

one or more computer programs encoded by said computer readable memory and configured to:

provide at least one declaration for an attribute to be handled as a real-time attribute associated with but external to a directory structure {¶0076};

receive a directory access protocol request for access to one or more attribute values from said associated directory structure {¶0090};

detect in said received request a request to access an attribute declared as a real-time external attribute {0091};

responsive to said detecting of a request for a real-time attribute, resolve a real-time value by obtaining an attribute value from a real-time source external to said directory structure {0092};

responsive to said resolving, converting said obtained attribute value from a real-time attribute to a static attribute, wherein said real-time attribute is incompatible with said directory access protocol, and wherein said static attribute is compatible with said directory access protocol {¶¶0087, 0093, 0137}; and

return to a requester said converted real-time attribute directly in said directory access protocol, wherein storing and updating of said converted real-time attribute value in said directory structure is eliminated or avoided {¶¶0076, 0087, 0101}.

And, independent Claim 23 is directed towards analogous system embodiments according to our invention:

23. A system comprising a hardware means for performing a logical process, wherein said logical process comprises:

providing at least one declaration for an attribute to be handled as a real-time attribute associated with but external to a directory structure {¶0076};

receiving a directory access protocol request for access to one or more attribute values from said associated directory structure {¶0090};

detecting in said received request a request to access an attribute declared as a real-time external attribute {0091};

responsive to said detecting of a request for a real-time attribute, resolving a real-time value by obtaining an attribute value from a real-time source external to said directory structure {0092};

responsive to said resolving, converting said obtained attribute value from a real-time attribute to a static attribute, wherein said real-time attribute is incompatible with said directory access protocol, and wherein said static attribute is compatible with said directory access protocol {¶¶0087, 0093, 0137}; and

returning to a requester said converted real-time attribute directly in said directory access protocol, wherein storing and updated of said converted real-time attribute value in said directory structure is eliminated or avoided {¶¶0076, 0087, 0101}.

Grounds for Rejection For Which Review is Sought per 37 CFR §41.37(c)(1)(vi)

Appellant respectfully requests review and reversal of:

(a) the rejections of claims 8, 12, 13, and 20 - 34 under 35 U.S.C. §112, first paragraph; and

(b) the rejections of claims 8, 12, 13, and 20 - 34 under 35 U.S.C. §103(a) over US Pre-Grant Published Patent Application 2002/0147857 to Sanchez, II *et al* (hereafter Sanchez) in view of US Pre-Grant Published Patent Application 2008/0086402 to Patel, *et al* (hereafter Patel), in view of US Pre-Grant Published Patent Application 2003/0120502 to Robb, *et al* (hereafter Robb).

Arguments per 37 CFR §41.37(c)(1)(vii)

Rejections under 35 U.S.C. §112, First Paragraph

Claims 8, 12, 13, and 20 - 34.

In these rejections, the Examiner has held that support is not found within the specification for the claim limitation of:

> ". . . responsive to said resolving, converting said obtained attribute value from a real-time attribute to a static attribute, wherein said real-time attribute is incompatible with said directory access protocol, and wherein said static attribute is compatible with said directory access protocol" and "returning to a requester said converted real-time attribute directly in said directory access protocol, wherein storing and updating of said converted real-time attribute value in said directory structure is eliminated or avoided . . . "

The wording of the rejection is quite succinct, so it is not entirely clear which part of this limitation the Examiner believes is unsupported by the specification. Considered in the context of the rationale provided in the rejections under 35 U.S.C. §103(a), Appellant believes the term which the Examiner holds is unsupported is "converting". If this is not true, Appellant respectfully requests that the Examiner clarify the reasons for rejection to the Board in the Examiner's Answer. If the Examiner fails to clarify, Appellant respectfully requests the Board to remand with instruction to clarify if the rejections are not reversed with the claims allowed.

Proceeding under this assumption, regarding ". . . responsive to said resolving, converting said obtained attribute value from a real-time attribute to a static attribute", we have asked the Examiner to consider that our disclosure states "resolving" prolifically throughou the disclosure (¶¶0064, 0067 - 0068, 0076, 0078, 0079, 0086 - 0086, 0093, 0098, 0100 - 0101, 0129, and the Abstract).

The Examiner seems to draw distinction between "resolving" and "converting" (see details of the rejections under 35 U.S.C. §103(a)) as being different actions, which Appellant

believes is an error in examination for failing to employ standard definitions to interpret the claims (emphasis added by Applicant):

> From Dictionary.com:
> **resolve** –*verb* (used with object)
>
> . . .
>
> 3. to reduce or **convert** by, or as by, breaking up or disintegration (usually fol. by to or into).
> 4. **to convert or transform by any process** (often used reflexively).
>
> . . .

> From the American Heritage Dictionary:
> **resolve** *(v)*
>
> . . .
>
> 4. To change or **convert**: My resentment resolved itself into resignation.
>
> . . .

So, clearly in ordinary usage, the term "resolve" which appears prominently throughout Appellant's disclosure is synonymous with "convert", depending on context. The context that Appellant has disclosed is provided with regard to *changing* a real-time (a.k.a. dynamic) variable value (Title, Abstract, ¶¶0049, 0050, 0058, 0059, etc.) into a static value (¶¶0064, 0065, 0068, 0069, 0076). And, examples of processes through which the real-time values are transformed (e.g. converted, resolved) are provided such as:

> [0092]. . . accesses one or more real-time data sources (54) such as web sites, data interfaces, etc., performs the necessary data manipulations (e.g. calculations, image processing, etc.), and returns the resulting data . . .

We believe, using the extrinsic definition of resolve and convert, and considering our disclosure at the many points where the terms are used and exemplified that one of ordinary skill

in the art would not see the distinction between "resolve" and "convert" that the Examiner is employing in the rationale for the rejections.

By failing to provide more explicit rationale for the rejections, and by failing to provide any evidence that controverts these extrinsic definitions, we believe the Examiner has erred in making this rejection.

Rejections under 35 U.S.C. §103(a)

Claims 8, 12, 13, and 20 - 34

All of these claims include the argued resolving and converting terms. The Examiner has held that Appellant only teaches that the prior art "converts", but not that the invention "converts". Examiner has refused to afford synonymity to the terms "resolve", which is recited many times in the description of the invention, with the term "convert". Appellant respectfully submits that this is an error in examination to refuse to employ ordinary meanings of claim terms.

Appellant has disclosed that the prior art converts dynamic values to static values, however only *before storing the values in the LDAP directory.* This sequence of steps incurs processing delays to perform an update command on the value in the LDAP directory, which our invention eliminates by converting the value followed by directly inserting the value in the return payload of the return message according to the protocol, *without storing the value in the LDAP directory:*

> [0087] It is important to note that the v**alue "82 F" was never actually stored in or retrieved from the LDAP directory**. . . .
>
> . . .
>
> [0093] . . . Dynamic data values, however, are not stored in the LDAP directory or file system, thereby minimizing the overhead impact of adding dynamic data to a directory which is optimized for static data storage. . . .

Elimination of a step or element may impart patentability:

> The inquiry here establishes that the present invention includes the inventor's elimination of the need for damping. Because that insight was contrary to the understanding and expectations of the art, the structure effectuating it would not have been obvious to those skilled in the art. *Carl Schenk, A.G. v Nortron Corp.,* 713 F.2d 782, 218 USPQ 698, 700 (Fed. Cir. 1983)

Appellant respectfully submits that the addition of the special indicator regarding which attributes are to be directed accessed from their source to be converted to static values and returned directly into the return message to the LDAP requester is evidence of foresight which does not exist in the cited art. Thus, the cited art does not provide an obvious modification to eliminate storing of their converted values in the LDAP directory.

Whereas Appellant has claimed the limitation ". . . wherein storing and updating of said converted real-time attribute value in said directory structure is eliminated or avoided", it is incumbent on the Examiner to show where in the cited art such a limitation is taught or suggested in order to establish a proper *prima facie* case of obviousness under 35 U.S.C. §103(a). Support for this claimed limitation is found at Appellant's paragraphs 0064, 0078 and 0101 in addition to the previously-mentioned paragraphs (emphasis added by Applicant):

> [0064] . . . This approach **eliminates** the need to store the dynamic information in the directory, . . .

> [0078]. . . By **avoiding updating the value** of the attribute when no client is requesting its value, our invention avoids unnecessary use of processor and communications bandwidth. . . .

> [0101] (e) returning (85) the resolved value to the requesting client while **suppressing or avoiding the storing of the actual value in the LDAP directory**;

In the reasons for the rejection of the independent claims, the Examiner has not stated any reference citation where ". . . wherein storing and updating of said converted real-time attribute value in said directory structure is eliminated or avoided" is explicitly taught or suggested. Examiner has pointed to Robb's disclosure at ¶¶0074 and 0076:

> [0074] In an exemplary embodiment, the Integration container 107 is supported by Enterprise Application Integration (EAI) technologies, which address integration of applications using messaging technology, including a message bus, a Workflow Manager (WFM), and a Rules Engine (RE). The messaging supports point-to-point and publish and subscribe type messaging with varying quality of service (QoS) or bandwidth levels. According to one feature of the present invention, the sequential rules of a process, or Workflow Management (WFM), are defined and stored centrally in the Rules Engine instead of embedded in code throughout the system. Work Flow Management automates the sequence of actions necessary to complete a unit of work. A specific work flow is often initiated when a rule is fired in a rules engine. The Rules Engine evaluates events using a rule base and, when the criteria for a rule is met, fires the rule; rule firing can result in the initiation of a specific work flow. Integration also includes version control of services within the execution environment and joining multiple directory servers by using a Meta directory.

In ¶0074, Robb mentions "integration containers", message buses, Workflow Managers, publish/subscribe, Quality of service, and Rules engines, but there is no mention of eliminating or avoiding anything. Appellant respectfully submits that the Examiner has not properly established what, if anything, in this paragraph has been eliminated or avoided, and what is being considered an LDAP update command that Appellant claims to avoid or eliminate from the prior art process. Looking closely at ¶0076, it still is not clear what Robb has supposedly eliminated or avoided:

> [0076] The Applications container 111 utilizes the Application Services 109 to deliver applications to the end-users. This Applications container 111 supports ecommerce relationships, such as business-to-business (B2B) and business-to-consumer (B2C). These relationships are supported with product catalogs for the presentation and management of products, shopping carts, order management, and order processing. Unified Messaging is also a part of the Applications container 111 and provides one common format for the transmission and storage of messages such as email, voice mail, fax, and video. The Applications container 111 also includes Content Management for acquiring and aggregating of static and dynamic content, such as news, weather, stock quotes, and syndicated feeds. Directory services are also supported by the Applications container 111; for example, corporate directories may be employed to allow the lookup of employee information based on various search criteria. The AIP, through the Applications container 111, can also support a variety of business applications; e.g., GREAT PLAINS, PEACHTREE, QUICKEN, MICROSOFT OFFICE, and QUICKBOOKS. These business applications are also examples of applications that

could be made available by ASPS.

There is no mention whatsoever in Robb's ¶0076 of avoiding or eliminating any steps of any processes. In fact, Robb appears to actually update the LDAP entries, not avoid updates, as shown in Figure 30 "update subscriber info", "store user profile updates", and "store device profile updates", as well as in ¶¶0198 and 0204 where updates are stored.

For these reasons, Appellant respectfully requests reversal of the rejections and allowance of the claims due to errors in examination including (a) failing to consider the claims in light of the specification regarding claim terms convert and resolve, (b) failing to use ordinary definitions of terms to interpret the claims, and (c) failing to establish a *prima facie* case of obviousness by failing to show where each and every claim term, element, and limitation of the claims is shown in the cited references. Further, in view of the very different terminology of the cited and relied-upon paragraphs of the Robb reference for the rejections from the terminology of Appellant's disclosure and claims, Appellant respectfully submits that (d) the Appellant's own disclosure has been erroneously read into the Robb reference.

Respectfully,

/ Robert H. Frantz /

Robert H. Frantz, Reg. № 42,553
Tel: (405) 812-5613
Franklin Gray Patents, LLC

Claims Appendix

per 37 CFR §41.37(c)(1)(viii)

Clean Form of Amended Claims

Claims 1- 7 (cancelled).

8. A method comprising:

providing at least one declaration for an attribute to be handled as a real-time attribute associated with but external to a directory structure;

receiving a directory access protocol request for access to one or more attribute values from said associated directory structure;

detecting in said received request a request to access an attribute declared as a real-time external attribute;

responsive to said detecting of a request for a real-time attribute, resolving a real-time value by obtaining an attribute value from a real-time source external to said directory structure;

responsive to said resolving, converting said obtained attribute value from a real-time attribute to a static attribute, wherein said real-time attribute is incompatible with said directory access protocol, and wherein said static attribute is compatible with said directory access protocol; and

returning to a requester said converted real-time attribute directly in said directory access protocol, wherein storing and updating of said converted real-time attribute value in said directory structure is eliminated or avoided.

Claims 9 - 11 (cancelled)

12. The method as set forth in Claim 8 wherein said detecting comprises parsing a Lightweight Directory Access Protocol requests for attribute values.

13. The method as set forth in Claim 8 wherein said returning comprises returning said value according to a Lightweight Directory Access Protocol.

Claims 14 - 19 (cancelled).

20. A computer readable memory comprising:

a computer readable memory suitable for encoding computer programs; and

one or more computer programs encoded by said computer readable memory and configured to:

provide at least one declaration for an attribute to be handled as a real-time attribute associated with but external to a directory structure;

receive a directory access protocol request for access to one or more attribute values from said associated directory structure;

detect in said received request a request to access an attribute declared as a real-time external attribute;

responsive to said detecting of a request for a real-time attribute, resolve a real-time value by obtaining an attribute value from a real-time source external to said directory structure;

responsive to said resolving, converting said obtained attribute value from a real-time attribute to a static attribute, wherein said real-time attribute is incompatible with said directory access protocol, and wherein said static attribute is compatible with said directory access protocol; and

return to a requester said converted real-time attribute directly in said directory access protocol, wherein storing and updating of said converted real-time attribute value in said directory structure is eliminated or avoided.

21. The computer readable memory as set forth in Claim 20 wherein said detecting comprises parsing a Lightweight Directory Access Protocol requests for attribute values.

22. The computer readable memory as set forth in Claim 20 wherein said returning comprises returning said value according to a Lightweight Directory Access Protocol.

23. A system comprising a hardware means for performing a logical process, wherein said logical process comprises:

providing at least one declaration for an attribute to be handled as a real-time attribute associated with but external to a directory structure;

receiving a directory access protocol request for access to one or more attribute values from said associated directory structure;

detecting in said received request a request to access an attribute declared as a real-time external attribute;

responsive to said detecting of a request for a real-time attribute, resolving a real-time value by obtaining an attribute value from a real-time source external to said directory structure;

responsive to said resolving, converting said obtained attribute value from a real-time attribute to a static attribute, wherein said real-time attribute is incompatible with said directory access protocol, and wherein said static attribute is compatible with said directory access protocol; and

returning to a requester said converted real-time attribute directly in said directory access protocol, wherein storing and updated of said converted real-time attribute value in said directory structure is eliminated or avoided.

24. The system as set forth in Claim 23 wherein said hardware means comprises at least in part a microprocessor.

25. The system as set forth in Claim 23 wherein said hardware means comprises at least in part an electronic circuit.

26. The system as set forth in Claim 25 wherein said electronic circuit is selected from a group comprising an application specific integrated circuit, and a programmable logic circuit.

27. The system as set forth in Claim 23 wherein said detecting comprises parsing a Lightweight Directory Access Protocol requests for attribute values.

28. The system as set forth in Claim 23 wherein said returning comprises returning said value according to a Lightweight Directory Access Protocol.

29. The method of Claim 8 wherein said resolving a real-time value by obtaining an attribute value from a real-time source external to said directory structure further comprises selecting according to a predetermined selection schema a real-time attribute processor from a plurality of available real-time attribute processors, invoking said selected real-time attribute processor, and wherein said resolving is performed by said invoked real-time attribute processor.

30. The method of Claim 29 wherein said predetermined selection schema comprises a schema employing a variation of a name of said requested directory attribute to identify a real-time attribute processor for selection.

31. The computer readable memory of Claim 20 wherein said resolving a real-time value by obtaining an attribute value from a real-time source external to said directory structure further comprises selecting according to a predetermined selection schema a real-time attribute processor from a plurality of available real-time attribute processors, invoking said selected real-time attribute processor, and wherein said resolving is performed by said invoked real-time attribute processor.

32. The computer readable memory of Claim 31 wherein said predetermined selection schema comprises a schema employing a variation of a name of said requested directory attribute to identify a real-time attribute processor for selection.

33. The system of Claim 23 wherein said logical process resolving a real-time value by obtaining an attribute value from a real-time source external to said directory structure further comprises a logical process selecting according to a predetermined selection schema a real-time attribute processor from a plurality of available real-time attribute processors, invoking said selected real-time attribute processor, and wherein said resolving is performed by said invoked real-time attribute processor.

34. The system of Claim 33 wherein said predetermined selection schema comprises a schema employing a variation of a name of said requested directory attribute to identify a real-time attribute processor for selection.

Evidence Appendix

per 37 CFR §41.37(c)(1)(ix)

No evidence has been submitted by applicant or examiner pursuant to 37 CFR §§1.130, 1.131, or 1.132.

Related Proceedings Appendix

per 37 CFR §41.37(c)(1)(x)

No decisions have been rendered by a court or the Board in the related proceedings as identified under 37 CFR §41.37(c)(1)(ii).